Universidade Federal de Alagoas (UFAL)
Campus Arapiraca

Programação Orientada a Objetos (POO)

10 - Exception

Alexandre de Andrade Barbosa

alexandre.barbosa@arapiraca.ufal.br

Objetivos

Objetivos

- Apresentar o conceito e motivar o uso de exceções

Exception
Exceções

- A linguagem Java possui um mecanismo para detectar situações anormais ao funcionamento do programa
- Exemplos de situações anormais em um programa são:
  - falha de memória;
  - impossibilidade de gravar em disco;
  - abrir um arquivo inexistente;
  - acessar um indice inválido em uma estrutura (ex. array);
  - outros
- Do ponto de vista de implementação uma exceção é uma classe que herda da classe Exception

Exception
Tipos de erros

- Erros de lógica de programação
  - ex.: acessar um indice inválido em uma estrutura (ex. array), divisão por zero, etc.
  - devem ser corrigidos pelo programador
- Erros devido a condições de execução
  - ex.: arquivo não encontrado, rede fora do ar, etc.
  - ocorrem de forma imprevisível, mas podem ser corrigidos em tempo de execução
- Erros graves
  - ex.: falta de memória, etc
  - não podem ser contornados pelo programador

Exception
Exceções

```
1 public static void main(String[] args) {
2     String s = JOptionPane.showInputDialog("Digite um numero:
          ");
3     int i    = Integer.parseInt(s);
4     JOptionPane.showMessageDialog(null, "Valor digitado " + i);
5 }
```

Exception
Exceções

- Podemos controlar os erros:
  - Capturando de exceções;
  - Jogando exceções;
  - Criando exceções;
  - Levantando exceções.

Exception

Exceções - Capturando exceções

```
1  public static void main(String[] args) {
2      try {
3          String s = JOptionPane.showInputDialog("Digite sua
               idade: ");
4          int i = Integer.parseInt(s);
5          JOptionPane.showMessageDialog(null, "A idade informada
               foi " + i);
6      } catch (NumberFormatException e) {
7          System.out.println("Formato de número inválido!");
8      } catch (Exception e) {
9          System.out.println("Um erro inesperado ocorreu!");
10     }
11 }
```

Exception

Exceções - Jogando de exceções

```
1  public class ExemploException {
2      public int leitura() throws NumberFormatException {
3          String s = JOptionPane.showInputDialog("Digite sua
               idade: ");
4          int i = Integer.parseInt(s);
5          return i;
6      }
7      public static void main(String[] args) {
8          try {
9              ExemploException ex = new ExemploException();
10             int i = ex.leitura();
11             JOptionPane.showMessageDialog(null, "A idade
                   informada foi " + i);
12         } catch (NumberFormatException e) {
13             System.out.println("Formato de número inválido!");
14         } catch (Exception e) {
15             System.out.println("Um erro inesperado ocorreu!");
16         }
17     }
18 }
```

Exception

Exceções - Criando exceções

```
1 public class IdadeInvalidaException extends Exception {
2
3     public IdadeInvalidaException() {
4         super("A idade informada é invalida!");
5     }
6
7 }
```

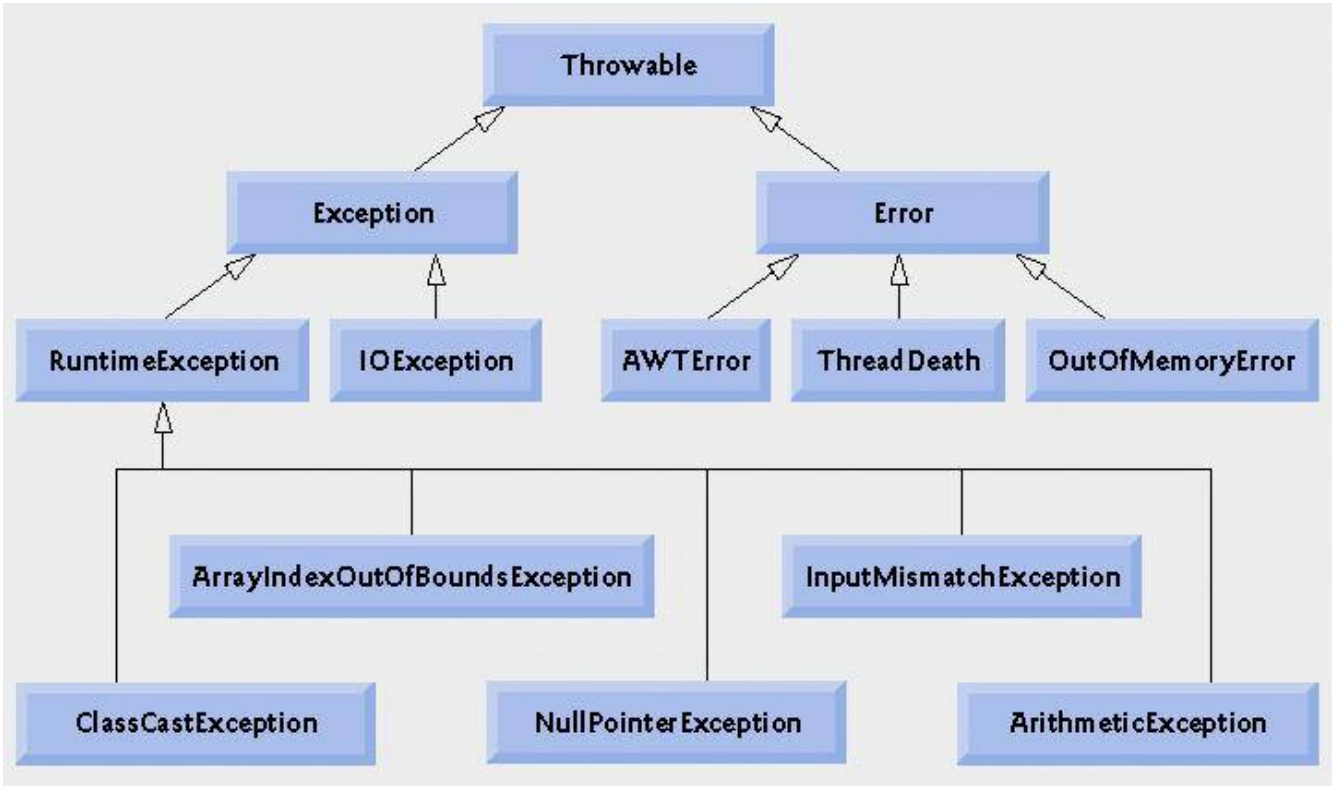
Exception

Exceções - Levantando exceções

```
1 public int leitura() throws IdadeInvalidaException {
2     String s = JOptionPane.showInputDialog("Digite sua idade:
      ");
3     int i = Integer.parseInt(s);
4
5     if (i < 0) {
6         throw new IdadeInvalidaException();
7     }
8     return i;
9 }
```

Exception
Hierarquia de exceções

- Uma exceção em Java herda (direta ou indiretamente) da classe **java.lang.Exception**



Exception
Hierarquia de exceções

- Exceções podem ser:
  - verificáveis
  - não verificáveis
- Classes de exceções verificáveis herdam de Exception
- Classes de exceções não verificáveis herdam de RuntimeException ou Error

Exception
Hierarquia de exceções

- Exceções verificáveis
  - o compilador verifica cada chamada de método e declaração de método para determinar se o método lança um exceção;
  - o compilador exige que se declare ou capture a exceção;
  - se não for capturada ou declarada, ocorrerá um erro.
- Exceções não verificáveis
  - o compilador não verifica o código para ver se a exceção foi capturada ou declarada;
  - se ocorrer e não tiver sido capturada, o programa terminará ou executará com resultados inesperados.

Exception
Vantagens no uso de exceções

- Algumas vantagens associadas ao uso de exceções
  - Separação de código de erro e código “normal”;
  - Propagação do tratamento de erros;
  - Agrupamento e categorização de erros;
  - Mensagens de erro padronizadas.

Exception
Exercícios

Exercício

1. Implemente exceções na aplicação bancária para as seguintes situações:
   - valor de saque inválido, para valores negativos
   - valor de depósito inválido, para valores negativos
   - saque não realizado, para valor de saque superior ao saldo
   - conta não encontrada, para um número de conta que não está entre as contas de um cliente

Exception
Resumo

Resumo

- Exceções são um mecanismo que possibilita controlar os erros que ocorrem em um programa
- Podemos controlar os erros:
  - Capturando de exceções;
  - Jogando exceções;
  - Criando exceções;
  - Levantando exceções.

Leituras recomendadas

Caelum

Java e Orientação a Objetos, 2011.

Capítulo 11: Controlando os erros com Exceções

David Barnes e Michael Kolling

Programação Orientada a Objetos com Java, 2008.

Seção 12.4: Princípios de lançamento de exceção

Seção 12.5: Tratamento de exceções

Seção 12.6: Definindo novas classes de exceção

Exception

Perguntas?

Alexandre de Andrade Barbosa
alexandre.barbosa@arapiraca.ufal.br